# The Application Protocol Information Base World Wide Web Gateway[*][†]

Joshua Lubell
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Building 220, Room A127
Gaithersburg MD 20899 USA
Email: `lubell@cme.nist.gov`

NIST Technical Report: NISTIR 5868

July 31, 1996

**Abstract**

The Application Protocol Information Base (APIB) is an on-line repository of documents for the *St*andard for the *E*xchange of *P*roduct model data (STEP, officially ISO 10303—Product Data Representation and Exchange). Document types in the APIB include STEP Application Protocols and Integrated Resources. Application Protocols are standards that are intended to be implemented in software systems, and Integrated Resources are used by them as building blocks. Application Protocols and Integrated Resources are represented in the Standard Generalized Markup Language (SGML) in the APIB in order to facilitate efficient information search and retrieval. This paper describes a World Wide Web gateway to the APIB, implemented using the Common Gateway Interface (CGI) standard. The APIB gateway allows STEP developers to efficiently search for ISO 10303 standards and supporting information. The only client software required to use the APIB gateway is a third party web browser.

*Keywords*: CGI; World Wide Web; STEP; SGML; Application Protocol; Integrated Resource; text retrieval; HTML; Tcl

[*]Trade names and company products are mentioned in the text in order to adequately specify experimental procedures and equipment used. In no case does this identification imply recommendations or endorsements by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

[†]Development of the APIB and the APIB gateway is funded by the National Institute of Standards and Technology's Systems Integration for Manufacturing Applications (SIMA) program, a part of the US government's High Performance Computing and Communications initiative.

## 1 Background

STEP[ISO1], the *St*andard for the *E*xchange of *P*roduct model data, specifies a description of product data throughout its life cycle in a computing platform-independent manner. An Application Protocol (AP) in STEP defines the information requirements for a particular area of design, manufacturing, engineering, or product support. APs provide a neutral representation for sharing product data among dissimilar software applications. Each STEP Integrated Resource (IR) defines a set of re-usable constructs that serve as building blocks for developing APs. These constructs are specified using an information modeling language called *EXPRESS*[ISO11]. APs are developed according to a rigorous methodology[PALMER] dictated by the STEP community. The end result of this development process is a standards document that specifies the AP's requirements and a technical solution for data exchange that uses STEP technology.

The STEP community is international in scope and includes participants from numerous industries and research institutions. Over twenty-five nations and more than 200 companies are currently involved in developing STEP. A typical STEP AP team has four or more people, typically not all working for the same company or even living in the same country. It takes this team, on the average, a year and a half to produce an initial AP specification for a typical engineering function. Thus AP development is both time-consuming and expensive.

There are two principal reasons why AP development is so cumbersome. One reason is the geographical and organizational dispersion in the STEP community. The other is the lack of an integrated STEP-tailored software environment for AP developers. To address these issues, NIST is building an Internet-accessible, integrated software tool suite in order to accelerate the AP development process and to improve AP quality. This software environment uses a central information registry, the AP Information Base (APIB)[LUB96].

The APIB is an on-line repository under development at NIST for users and developers of STEP. The APIB currently contains only STEP IRs. When it is complete, the APIB will also contain APs, other STEP documents describing representation and implementation methods, and additional information useful to the STEP community such as schedules, issue logs, and issue resolutions. The APIB's contents are indexed for efficient access using a text retrieval engine[OTC]. Each document in the APIB is stored in its native format. Translation for viewing or publishing purposes is done on demand. APs and IRs are represented in the Standard Generalized Markup Language (SGML)[GOLD], a language for describing the structure of documents for use in text processing applications[1]. AP and IR documents are marked up using SGML document type definitions (DTDs) that have been developed specifically for the requirements of STEP[PHIL]. Other documents will be represented as plain ASCII text, or in SGML using DTDs less complex than those for APs

[1]SGML is defined in the international standard ISO 8879.

and IRs. The text retrieval engine uses the SGML markup in the APs and IRs to index their data. Thus users can issue queries to the APIB referring to the SGML structures specified for STEP.

This paper describes a World Wide Web interface to the APIB. This implementation makes use of the Common Gateway Interface (CGI) standard for interfacing external applications with web servers[BLEE95]. CGI scripts generate HTML dynamically in response to requests for web pages. The APIB gateway uses HTML forms to obtain input data from the user. The input is then processed by the APIB gateway's CGI scripts[2]. The APIB gateway provides APIB access services to STEP AP developers through their web browser software, eliminating the need for AP developers to install APIB client software. It also eliminates the need for NIST to build and maintain separate APIB user interfaces for multiple computer platforms.

## 2 Organization of the APIB Gateway

The APIB gateway provides an interface between a user's web browser and the APIB's data access services. This interface consists of a collection of CGI scripts implementing various data access services and an SGML-to-HTML translator for STEP as shown in Figure 1. Each CGI script answers a particular type of user request. The requests are issued by means of entering information in a form and pressing a button to submit the request. The CGI script triggered by the form issues a query to the APIB and, using the query result, generates a new HTML page to display to the user. The new HTML page typically contains another form soliciting user input. The sample user session in Section 3 depicts the relationship between forms, scripts, and the APIB.

The SGML-to-HTML translator is used to convert SGML-tagged data from the source documents in the APIB into HTML so they can be displayed in a web browser. The translation is only necessary for query results containing actual text from the APIB documents. If the query result does not contain an actual piece of the document (for example, the result could be a list of object names satisfying the query), then the CGI scripts generate HTML output without use of the translator. The bidirectional arrow between the CGI scripts and the APIB and the unidirectional arrows from the APIB to the translator and from the translator to the CGI scripts in Figure 1 indicate this conditional use of the translator.

[2]World Wide Web gateways often use HTML forms in this way as a means of obtaining input for CGI programs.

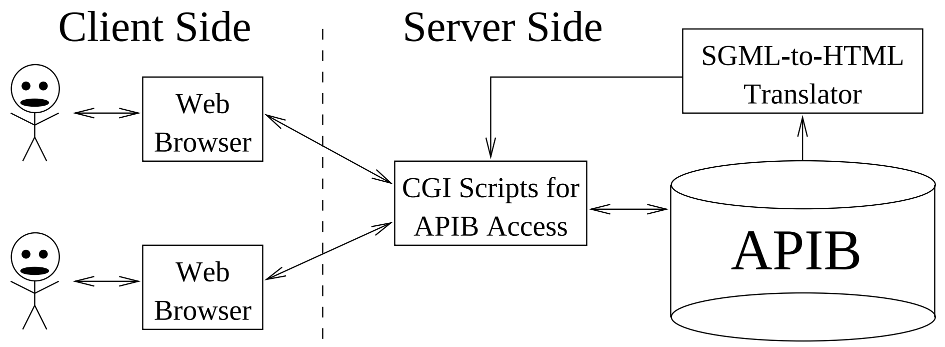


Figure 1: Architecture of the APIB gateway.

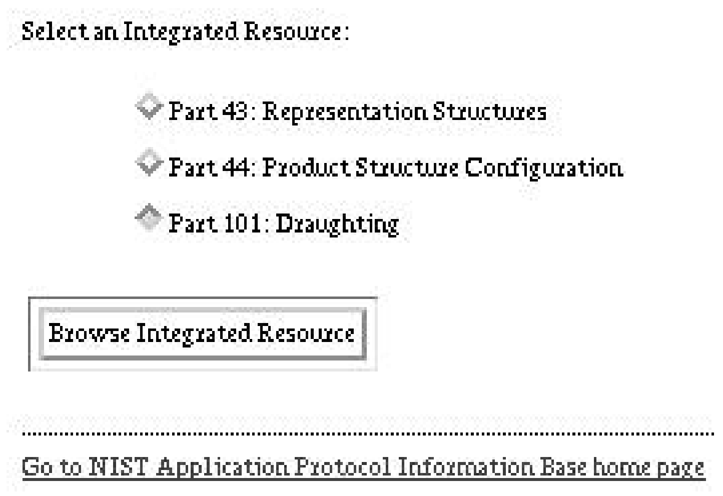
## 3 A Tour of the APIB Gateway

The best way to show how the APIB gateway works is through sample user sessions involving two of the services supported by the CGI scripts in Figure 1: an APIB browser and an IR Query interface. The browser provides a way for the user to view STEP documents using context-sensitive hypertext links. IR Query permits the user to search for certain object specifications in the IRs by specifying a name substring.

### 3.1 The APIB Browser

Since IRs contain re-usable building blocks for constructing APs (as was discussed in Section 1), STEP AP developers frequently need to search existing IR documents for information relevant to their industrial application. These documents have traditionally been available in paper form or electronically as PostScript, ASCII, or word processor-specific files. None of these formats provide a convenient way for AP developers to search for concepts across all of the existing IRs. They also fail to represent the IR document structure or the relationships between EXPRESS constructs in the IRs. Therefore, the traditional media for disseminating IR documents are not very useful to STEP AP developers. The APIB gateway's browser provides a way of disseminating IRs that should be far more useful to AP developers than either paper or the usual electronic formats.

Suppose the user chooses to view IRs using the APIB browser. A CGI script is invoked and generates a web page as shown in Figure 2. This web page contains a form consisting of a set of radio buttons for each IR in the APIB and a `Browse` button. The web page also contains a link back to the APIB home page. This form allows the user to choose one of


Select an Integrated Resource:

- Part 43: Representation Structures
- Part 44: Product Structure Configuration
- Part 101: Draughting

Browse Integrated Resource
Go to NIST Application Protocol Information Base home page

Figure 2: Form for choosing IR to browse.

the IRs in the APIB to browse. The CGI script obtained the names of the IRs in the form by querying the APIB.

Now suppose the user chooses `Part 101:  Draughting`[ISO101] and clicks the `Browse` button. Another CGI script will generate a web page as shown in Figure 3. This web page contains a form for choosing a section of Part 101 to browse. The form contains a collection of radio buttons corresponding to the major sections of a STEP IR document. The first four radio buttons represent sections whose names are common to all IRs. The three schema names, however, are specific to Part 101 and had to be obtained by means of an APIB query. An IR document also contains several appendices, but the APIB Gateway does not yet support browsing them. This capability will be added in the near future.

When the user clicks `Browse` in the form in Figure 3, a CGI script generates a web page containing the text of the IR document section chosen. This is accomplished in the following three steps:

1. The CGI script queries the APIB to retrieve the desired text.
2. The SGML-to-HTML translator converts the raw SGML data to HTML.
3. The CGI script uses the translated data to build a web page containing the desired section text along with any necessary forms, headers, and footers.

Note that this is the first time in the sample user session where the SGML-to-HTML

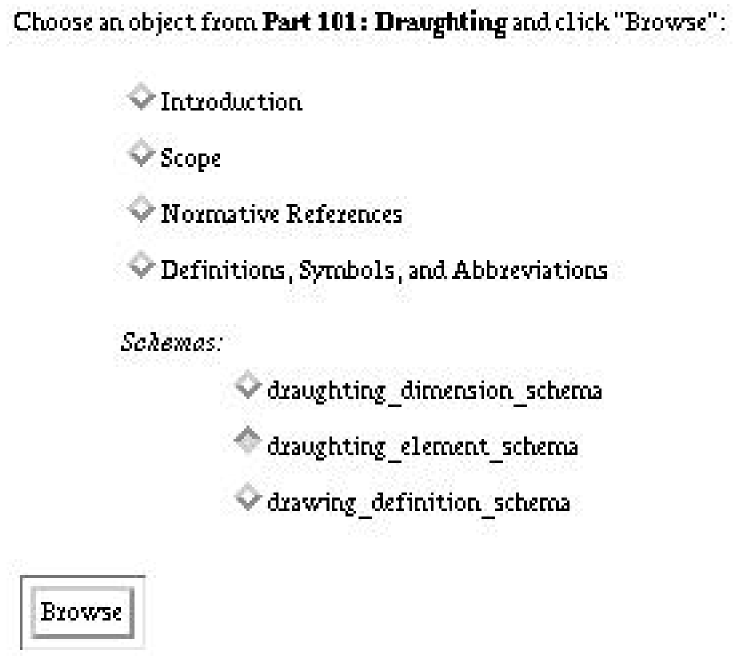
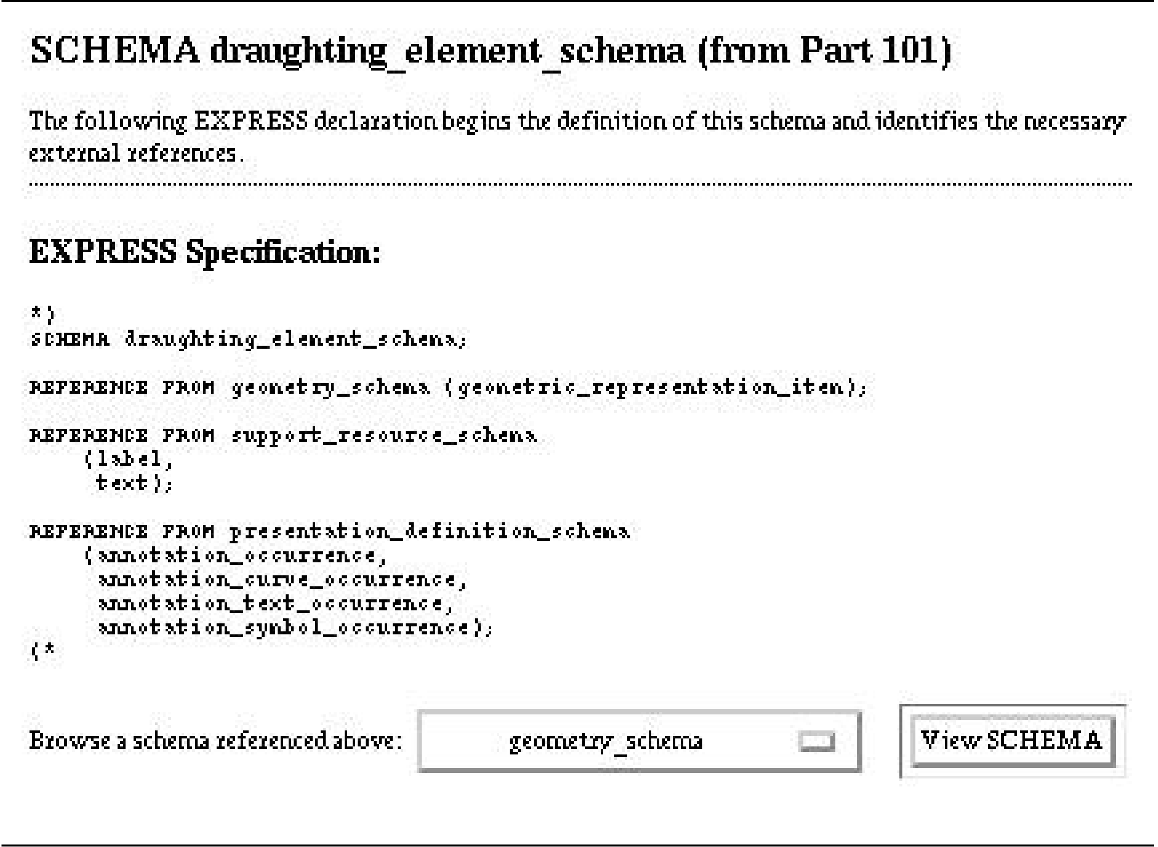

Choose an object from **Part 101: Draughting** and click "Browse":
Introduction
Scope
Normative References
Definitions, Symbols, and Abbreviations
*Schemas:*
draughting_dimension_schema
draughting_element_schema
drawing_definition_schema
Browse

Figure 3: Form for choosing section of IR Part 101 to browse.

translator was used. Heretofore, all of the APIB queries resulted in lists of names rather than raw SGML data.

Suppose now that the user chooses to view the `draughting_element_schema` from among the choices given in the form in Figure 3. The resulting web page that gets generated represents the content of this schema. This web page begins with an EXPRESS declaration for the schema. Figure 4 shows the EXPRESS declaration for the `draughting_element_schema`. This schema contain external references to objects in three other schemas: `geometry_schema`, `support_resource_schema`, and `presentation_definition_schema`. To make it easy for the user to view these referenced schemas, the web page includes a form next to the EXPRESS declaration with a pull-down list containing the schemas referenced. These schema names were obtained by querying the APIB for the schemas referenced by `draughting_element_schema`. The user can choose a schema name from the pull-down list and click a button to view that particular schema.

In addition to an EXPRESS declaration, the web page contains other documentation about the schema such as introductory text, fundamental concepts and assumptions, and some definitions of terminology used. These items (not shown in Figure 4) correspond to



## SCHEMA draughting_element_schema (from Part 101)

The following EXPRESS declaration begins the definition of this schema and identifies the necessary external references.

### EXPRESS Specification:

```
*)
SCHEMA draughting_element_schema;

REFERENCE FROM geometry_schema (geometric_representation_item);

REFERENCE FROM support_resource_schema
    (label,
     text);

REFERENCE FROM presentation_definition_schema
    (annotation_occurrence,
     annotation_curve_occurrence,
     annotation_text_occurrence,
     annotation_symbol_occurrence);
(*
```

Browse a schema referenced above: geometry_schema View SCHEMA

Figure 4: Portion of schema web page specifying EXPRESS declaration.

subsections of the schema's clause in the IR document.

The bulk of the schema's content, however, consists of the types, entities, functions, and rules that make up an EXPRESS model. The user can view any of the schema's objects by selecting the object name from a pull-down list in a form on the web page. The `draughting_element_schema` has entities and types, but no rules or functions. Therefore, this schema has the two forms for viewing objects as shown in Figure 5.

Suppose the user chooses to view the entity `dimension_curve_directed_callout`. A CGI script generates a web page containing the portion of the IR describing that entity. To obtain this information, the CGI script issues an APIB query for the entity definition for the entity named "dimension_curve_directed_callout". Since no two entities in STEP are allowed to have the same name, the APIB query result is a single entity definition clause from `draughting_element_schema`. Because the entity definition contains SGML-tagged text from the APIB, the query result must be run through the SGML-to-HTML translator so it can be displayed as a web page. Figure 6 shows the beginning of this web page. The web page begins with a definition of the entity, followed by a link to a figure, followed by

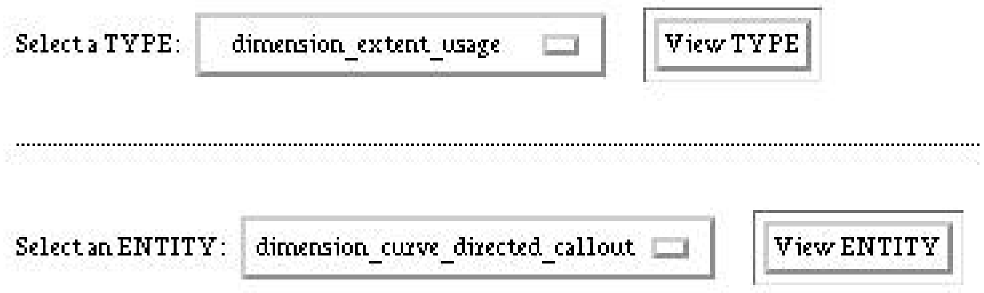


Figure 5: Forms for choosing entities and types for `draughting_element_schema`.

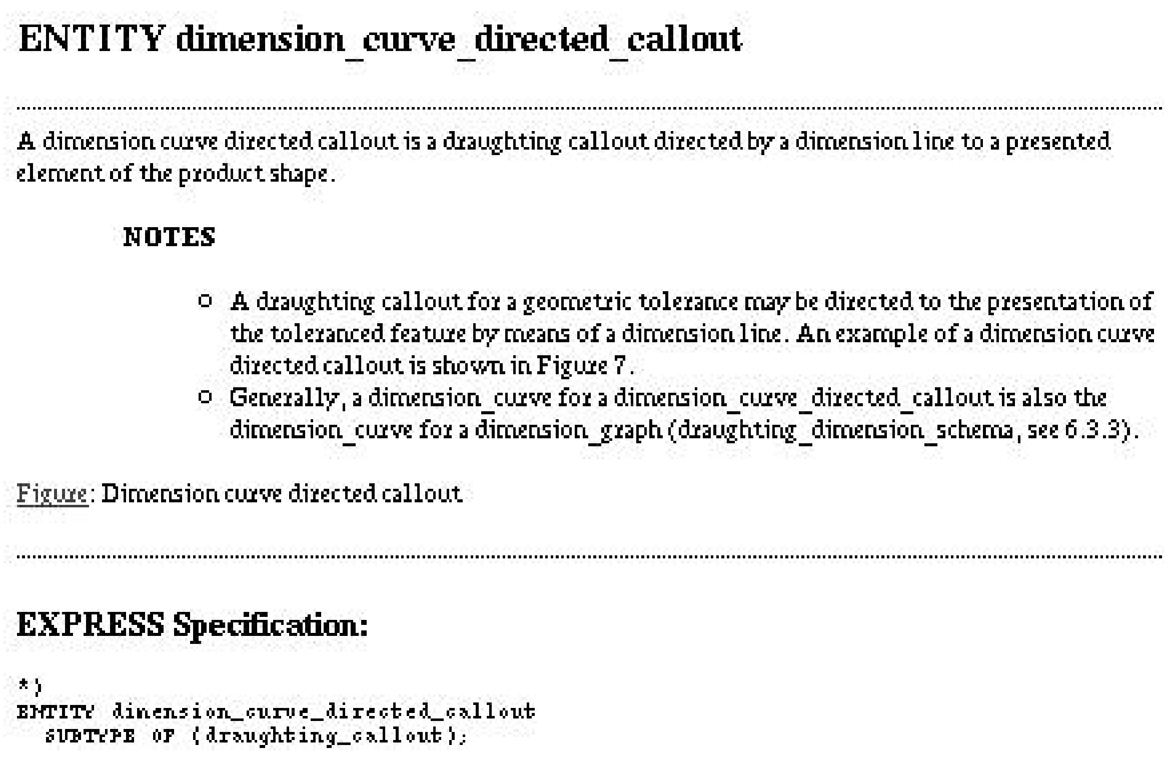
an EXPRESS specification (partially shown in Figure 6)[3]. Clicking on the figure link spawns a viewer to view the graphic. The CGI-generated web page also contains informative text not shown in Figure 6.

## 3.2 IR Query

STEP APs contain two classes of information models. The first class represents the AP's information requirements from a user's point of view. The second class describes the information requirements from a system integrator's point of view and is represented as an EXPRESS schema. This EXPRESS schema is created by selecting applicable constructs from the IRs and specializing them as needed by creating additional attributes or adding new constraints[BARN]. Mapping the concepts from the user's view to the appropriate IR constructs is a challenging task for AP developers because it requires considerable study and understanding of the IRs. The APIB gateway's IR Query interface provides a way to quickly retrieve IR constructs by specifying name substrings, thereby helping AP developers meet this challenge.

The form for issuing an Integrated Resource query, shown in Figure 7, contains a pull-down list of object types and a text entry field for specifying a search string. The object types available are `schemas`, `entities`, `types`, `rules`, and `functions`. Suppose the user is looking for an entity in the IRs to match a particular requirement, say a concept involving some kind of dimension. The user chooses `entities` from the pull-down list, enters `dimension` as the search string, and presses the `Search` button. The IR Query form's CGI script issues a query to the APIB for all entity definitions where the entity

[3]Note that the references to Figure 7 and clause 6.3.3 in the *NOTES* are currently hard coded in the raw SGML used to generate this web page (see Figure 10). Although the IR DTD supports cross references, the IRs in the APIB do not currently take advantage of this support. These IRs will be re-tagged in the near future to use the DTD's constructs for cross referencing.


ENTITY dimension_curve_directed_callout

A dimension curve directed callout is a draughting callout directed by a dimension line to a presented element of the product shape.

NOTES

- A draughting callout for a geometric tolerance may be directed to the presentation of the toleranced feature by means of a dimension line. An example of a dimension curve directed callout is shown in Figure 7.
- Generally, a dimension_curve for a dimension_curve_directed_callout is also the dimension_curve for a dimension_graph (draughting_dimension_schema, see 6.3.3).

Figure: Dimension curve directed callout

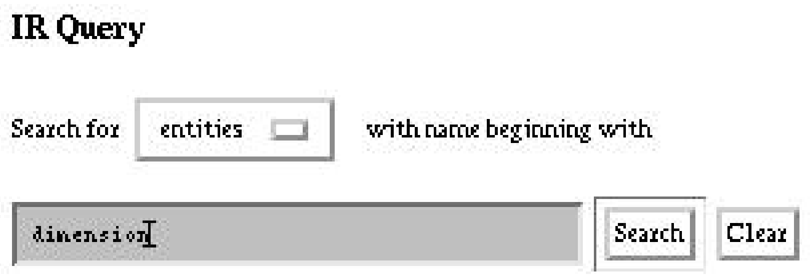
EXPRESS Specification:

```
*)
ENTITY dimension_curve_directed_callout
  SUBTYPE OF (draughting_callout);
```

Figure 6: Beginning of web page for entity `dimension_curve_directed_callout`.

name begins with "dimension". It then uses the query result to build a web page containing a form from which the user can choose an entity definition to view from a pull-down list of all the entity names. This web page, shown in Figure 8 also contains a link back to the IR Query form in case the user is unsatisfied with the list of results and wants to make another query.

If the user chooses an entity name from the pull-down list on the form in Figure 8 and presses the `View ENTITY` button, the same CGI script that created the entity definition web page in Figure 6 will create a web page for the entity chosen. At that point, the user will be in the APIB browser and can proceed in the same manner as if she had obtained the entity definition web page through pure browsing. Similarly, a user can use IR Query as a shortcut to a schema, type, function, or rule web page.

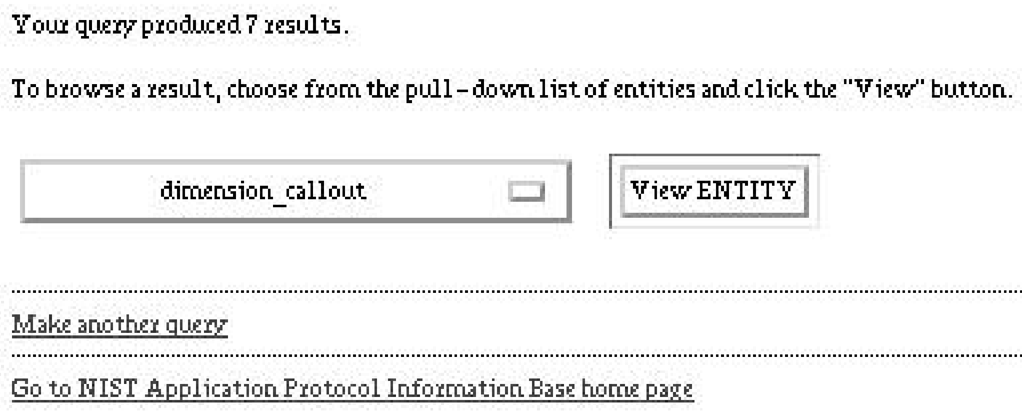


IR Query

Search for entities with name beginning with

dimension

Figure 7: Form for issuing an Integrated Resource query.



Your query produced 7 results.

To browse a result, choose from the pull-down list of entities and click the "View" button.

dimension_callout

Figure 8: Generated web page providing form for choosing a query result to view.

## 3.3 Other APIB Gateway Features

Section 3.1 illustrated how users can view the APIB contents at a very high level as collections of document types, choose a particular document type (IR), then view a specific document (Part 101), and then view individual portions of that document. Using a tree metaphor for the APIB, one can think of this as starting at the root node and traversing the APIB's search space by visiting successively higher leaves. Just as important is the ability to navigate backwards, that is from a more specific view of the APIB contents to a less specific, higher level one. For example, a user viewing a schema object might want to go back to the schema web page and choose another object from that schema, or a user might want to look at an IR's scope after viewing a schema from the IR. In order to facilitate traversal from a narrower, more detailed view to a broader, less detailed view, all of the web pages generated by the APIB Gateway's CGI scripts contain buttons near the bottom of the page for backtracking. These buttons provide hyperlinks from the current web page to all nodes in the APIB leading up to the root node[4]. When a user clicks on a backtracking button, an APIB query is issued to generate the appropriate web page.

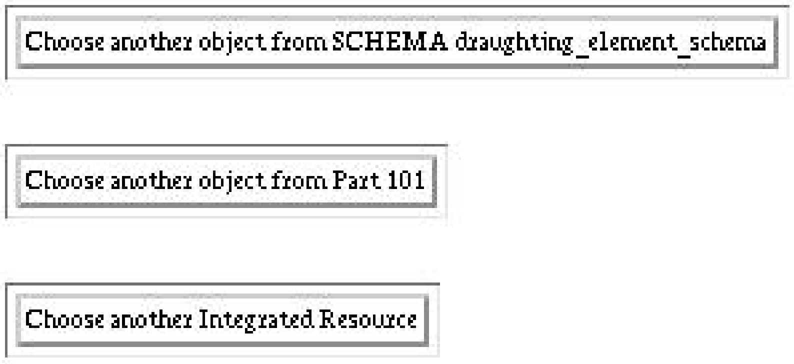

Choose another object from SCHEMA draughting_element_schema
Choose another object from Part 101
Choose another Integrated Resource

Figure 9: Backtracking buttons for entity `dimension_curve_directed_callout`.

As an example, consider the web page for the entity `dimension_curve_directed_callout`. Figure 6 showed the beginning of this web page. Figure 9 shows the backtracking buttons for this web page. The first button provides a link to the schema containing this entity. The second button links to the entity's IR Part. The third button links to the web page for choosing from among the IRs.

Another important capability is the ability to extract raw SGML text from the APIB. Since STEP AP developers re-use IR constructs as building blocks, they need to have a way to include portions of IR documents in their AP. For example, suppose an AP developer building an AP's EXPRESS schema wants to use the entity `dimension_curve_directed_callout`. Rather than rewrite an entity definition, the AP developer can obtain the SGML source for `dimension_curve_directed_callout`'s web page by clicking on a `View SGML Source` button on the web page. A CGI script is then invoked which produces a web page containing the raw SGML for the entity definition. Figure 10 shows a portion of this web page. The AP developer can then extract the SGML from the APIB by using the web browser's "Save" command[5].

[4]If the web browser caches previously viewed web pages, users can achieve the same effect by using their web browser's backtracking mechanism. They will achieve a faster response because the APIB will not be queried. However, if the APIB has recently been modified, the previously viewed web pages will be out of date. Also, if the user obtained the current web page by choosing a name from a list of IR Query results, then there are no cached pages to backtrack to, and the user has no choice but to use the backtracking push buttons.

[5]This would be improved if users could review the SGML, leave a pointer to it, and invoke a copy at AP publication time. This would require maintaining a table of pointers on the APIB server for each user. These tables could be maintained in a manner similar to the method the NIST EXPRESS Server[LIBES94] (see Section 6) uses to maintain users' private directories.

```
<entity.description>
A dimension curve directed callout is a draughting callout directed
by a dimension line to a presented element of the product shape.
<note.group>
<note.in.group linkid="note">
A draughting callout for a geometric tolerance may be
directed to the presentation of the toleranced feature by means
of a dimension line.  An example of a dimension curve directed
callout is shown in Figure 7.
</note.in.group>
<note.in.group linkid="note">
Generally, a dimension_curve for a dimension_curve_directed_callout
is also the dimension_curve for a dimension_graph
(draughting_dimension_schema, see 6.3.3).
</note.in.group>
</note.group>
<figure object="p101fig7" caption="Dimension curve directed callout">
</entity.description>
```

Figure 10: Portion of raw SGML data for entity `dimension_curve_directed_callout`.

## 4 APIB Gateway Implementation

The APIB gateway's CGI scripts are written in Tcl[OUST], a scripting language for controlling and extending applications. Tcl was chosen as the CGI programming language mainly because the Tcl language is flexible, portable, easy to use, and because several extensions useful to the APIB have already been implemented. The Tcl extensions the APIB gateway uses are:

- A command for tokenizing and scanning output from the APIB's retrieval engine[LUB95].
- Commands for controlling the APIB's retrieval engine and making queries[LUB96].
- A CGI library for generating HTML and passing data from HTML forms to CGI scripts[LIBES96].

This section illustrates how the APIB gateway was implemented, using a single CGI script — the script for creating schema object web pages — as an example.

An APIB gateway CGI script has three basic tasks to perform. First, it must assemble all of the input data. Some of this input data is entered in HTML forms by the user. Other input data is state information which must be passed to the script because

HTTP[BLEE96], the communication protocol used by web servers and clients, is stateless. Each HTTP request/response chain has no knowledge of previous HTTP communications. Therefore, it is up to the APIB gateway implementation to keep track of any necessary history information. Fortunately, HTML provides a way to accomplish this using HIDDEN fields in forms. HIDDEN fields contain values that get passed to CGI scripts, but they are invisible to the user.

To illustrate how the APIB gateway keeps track of the APIB browser's state, consider the CGI script that generated the web page in Figure 6. In order for this web page to provide the backtracking functionality shown in Figure 9, the CGI script must be aware of how the entity `dimension_curve_directed_callout` fits into the overall structure of the APIB. Specifically, the CGI script needs to know `dimension_curve_directed_callout`'s schema, the STEP document containing this schema, and the document type being browsed. The following Tcl code obtains this information:

```
cgi_import doctype
catch {cgi_import ir}
catch {cgi_import irsec}
```

The `cgi_import` command, part of the CGI library, retrieves an input value from a form. The last two calls to `cgi_import` are enclosed in `catch` commands in order to prevent the CGI script from exiting if either `$ir` or `$irsec` are undefined. After the commands above are executed, the CGI script has access to information concerning the type of the document to which the entity belongs and, if available, the entity's STEP Part number and schema. The CGI library also has a command, `cgi_export`, for specifying HIDDEN values in a form. In order for a variable to be imported into a CGI script, it must either have a value obtained from input to the form that invoked the CGI script, or it must have been exported from the form that invoked the CGI script.

The second task an APIB gateway CGI script must perform is to query the APIB for the information it needs. This is done in the CGI script that generated the web page for `dimension_curve_directed_callout` as follows:

```
start_pat
set qresult [express_definition [set $expitem] $expabbrev]
if {![info exists irsec]} {
    set irsec [schema_name [set $expitem] $expabbrev]
    if {![info exists ir]} {
set ir [lindex [part $irsec] 0]
    }
}
quit_pat
```

This code uses several commands from the Tcl extension for controlling and making queries to *Pat*[OTC], the APIB's retrieval engine. `start_pat` starts up Pat and sets up a communications pipeline between Pat and the APIB. `quit_pat` exits Pat and closes the communications pipeline. `express_definition`, `schema_name`, and `part` are commands which perform APIB queries. `express_definition` returns the raw SGML text from the APIB describing a schema object. `schema_name` returns the name of the schema to which the object belongs. `part` returns a schema's STEP Part name and Part number. All of these APIB Tcl commands make use of the Tcl extension for tokenizing and scanning output from Pat.

If this CGI script was invoked by means of the user clicking a "View" button on a form from a schema web page (as in Figure 5), `$irsec` and `$ir` will contain whatever query result values they were assigned in the schema web page's CGI script. On the other hand, if this CGI script was invoked though the IR Query form (see Figure 7), `$irsec` and `$ir` will be undefined because IR Query's CGI script does not compute these values. Therefore, the Tcl code above tests to see if these variables are undefined and, if they are, queries the APIB to obtain the schema name and Part number.

The argument `[set $expitem]` passed to the `express_definition` and `schema_name` commands specifies the name of a schema object. `$expitem` has as its value the schema object type and contains one of the following four strings: "entity", "type", "function", "rule". Its value is determined by the context under which the CGI script is invoked[6]. This value is used to construct the name of the SGML element to pass to Pat. In the case of `dimension_curve_directed_callout`, `$expitem`'s value is "entity", and `[set $expitem]`'s value is "dimension_curve_directed_callout".

The CGI script's third and final task is to create a web page using the input from forms and the APIB query results. Much of this involves using Tcl commands in the CGI library for creating HTML elements such as headings, paragraphs, forms, and so on. However, the CGI script must invoke the APIB gateway's SGML-to-HTML translator in order to convert the raw SGML data in `$qresult` into HTML.

The translator is implemented using NSGMLS, an application of the SP[CLARK] SGML parser, and SGMLSpm[MEGG], a perl[VROM] class library for processing the output from NSGMLS. NSGMLS validates the SGML data and converts it into a format which a structure-controlled conforming SGML application can act upon. An application of SGMLSpm called "sgmlspl" uses the output from NSGMLS to build a set of objects representing the structure of the SGML data as defined by its DTD. sgmlspl takes as a second argument a specification file. The specification file used for the APIB gateway contains perl instructions for converting every SGML construct that can appear in an

[6]Actually, there are four CGI script names who share the same Tcl file by means of symbolic linking. Tcl code at the beginning of the file derives `expitem` from the value of `argv[0]`. Also, there are some additional implementation details that have been omitted from this discussion.

APIB query result into HTML. This specification file, although fairly long, was simple to write. The following excerpt from the specification file shows how the ENTITY.DEF element from the DTD is translated to HTML. When the translator encounters the beginning tag for ENTITY.DEF, it obtains the value of ENTITY.DEF's NAME attribute and outputs an HTML heading with the name of the EXPRESS entity. The translator performs no action when it encounters ENTITY.DEF's ending tag.

```
sgml('<ENTITY.DEF>', sub{
    my ($element,$event) = @_;
    my ($entname) = $element->attribute(NAME)->value;
    output "<h1>ENTITY " . $entname . "</h1>";
});
sgml('</ENTITY.DEF>', "");
```

The following Tcl procedure, `render_html`, invokes the translator and returns the HTML-tagged result. `render_html` takes three arguments: the raw SGML data to be translated, an argument used to identify the SGML document type, and the name of the raw SGML data's top level element. For example, to generate HTML for `dimension_curve_directed_callout`'s entity definition, `render_html` would be called with the entity definition's SGML fragment (obtained by querying the APIB), `ir`, and `ENTITY.DEF`.

```
proc render_html {qresult parttype doctype} {

# Uses an SGML parser, the SGMLSpm perl library, and an sgmlspl spec
# file to translate raw SGML data to HTML

    global env

    # get the SGML declaration

    set f [open $env(SGMLDEC)]
    set sgmldoc [read $f]
    close $f

    # append the appropriate DTD fragment to the SGML declaration.
    # The QUERY.RESULT element is needed to capture the general
    # inclusions (such as figures, examples, etc.) declared in the
    # content models for the AP and IR top level elements.

    append sgmldoc "<!DOCTYPE QUERY.RESULT \["
    if {[string compare $parttype ap] == 0} {
        append sgmldoc "<!ENTITY % apdtd PUBLIC \"-//NIST//DTD Publish - "
        append sgmldoc "STEP Application Protocols V 2.1//EN\">"
        append sgmldoc "%apdtd;"
    } else {
        append sgmldoc "<!ENTITY % irdtd PUBLIC \"-//NIST//DTD "
```

```
            append sgmldoc "STEP Integrated Resources V 2.1//EN\">"
            append sgmldoc "%irdtd;"
        }
        append sgmldoc "<!ELEMENT Query.Result - - (${doctype}) "
        append sgmldoc "+(%General.Inclusions)>"
        append sgmldoc "\]>"

        # wrap QUERY.RESULT tags around the SGML-tagged query result,
        # append it to the SGML declaration and DTD, parse the whole
        # thing, and pipe the Element Structure Information Set to
        # sgmlspl.

        append sgmldoc "<query.result>" $qresult "</query.result>"
        safe_exec result $env(BIN_DIR)/$env(PARSER) <<$sgmldoc | \
        $env(BIN_DIR)/sgmlspl ./$env(TRANSLATOR)

        puts $result
    }
```

The bulk of the code in `render_html` assembles an *SGML document* to send to the translator. A complete SGML document consists of the following:

- An *SGML declaration* describing basic facts about the dialect of SGML being used. `render_html` reads in the SGML declaration from a file.
- A DTD. In lieu of an actual DTD, SGML documents often contain a DOCTYPE declaration referring to an external DTD. `render_html` supplies a reference to either the IR or AP DTD, depending on `$parttype`'s value. `render_html` supplements the DTD with an additional enclosing element, `QUERY.RESULT`, whose purpose is to enable the SGML parser to recognize elements specified in the DTD as *general inclusions*. `%General.Inclusions` is defined in the IR and AP DTDs to refer to all elements that can appear in multiple places in STEP documents such as examples, notes, cross references, etc. Such "floating elements" are often specified in a DTD as general inclusions in order to avoid having to specify them explicitly in the content model for every element within which they might appear.
- The *document instance*. This is the document text marked up according to the DTD. The document instance in `render_html` is the raw SGML data contained in the `qresult` argument with `QUERY.RESULT` tags wrapped around it.

The instruction beginning with `safe_exec` runs the SGML document through NSGMLS and pipes the output to sgmlspl. `$env(TRANSLATOR)` is the sgmlspl specification file. `safe_exec`, a wrapper for the `exec` Tcl command, saves the result in the variable `result` and catches any error codes.

## 5 Related CGI Applications

The APIB gateway has two principal characteristics setting it apart from most other CGI applications:

- It serves a group of individuals developing, maintaining, and implementing an international standard, namely STEP. This is noteworthy because standards developers traditionally have had very little software support besides generic tools such as word processors and electronic mail[7].
- Its data is represented in SGML, and users can issue queries referring to structures defined in its DTDs. Because the STEP DTDs are highly detailed, the APIB gateway supports a rich variety of structured queries and hypertext linking.

Two other CGI applications sharing some of the APIB gateway's characteristics are the National Standards Systems Network (NSSN)[LASK] and a gateway for accessing humanities texts[PWIL] from university libraries. Another related project is the Digital Library Initiative at the University of Illinois[SCHATZ].

### 5.1 The NSSN

The NSSN, like the APIB gateway, is a service for obtaining standards information over the Internet. When completed, the NSSN will provide an electronic infrastructure linking numerous databases of standards resources throughout the United States. An NSSN testbed implementation has been set up for developing and testing the concepts and technologies to be used in actual NSSN production environment. The NSSN testbed currently allows users to search collections of sample documents using either the popular WAIS search engine or PRISE, a full text retrieval system being developed at NIST that uses fast and space-efficient indexing and searching algorithms[ROGERS]. Traditional text query languages have a syntax based on Boolean logic. WAIS and PRISE, on the other hand, allow unstructured natural language queries. The PRISE search engine uses a statistical ranking technique to retrieve the documents in the data set that best match the query. Studies have shown that users often get satisfactory results faster with an information retrieval system supporting natural language queries with statistical ranking than than they would using a Boolean-based system[HAR]. Efforts are underway to extend the NSSN testbed implementation to automatically determine from a query which standards database to search. The testbed's developers have also implemented a "web crawler" tool for maintaining and indexing hyperlinks to standards resources on the web.

[7]Although standards bodies such as ISO and ANSI may have elaborate publishing systems, they are often unavailable to those who actually write the standards.

The NSSN and the APIB gateway are similar in that they both provide a means for users to quickly obtain useful standards information. They are different in their scopes and in their levels of granularity. The APIB's scope is limited to STEP while the NSSN is general in nature. Unlike the APIB gateway, whose architecture is tied to the STEP DTDs, the NSSN's architecture can accommodate any electronically available standards resource. However, the NSSN's generality comes at a price. Because it does not assume any particular predefined structure for its standards documents, its architecture cannot easily support "STEP-centric" search interfaces such as the APIB Browser and IR Query interface discussed in Section 3.

Full text search engines such as the PRISE system used in the NSSN prototype could be useful in the APIB. PRISE's support for natural language queries can potentially provide an attractive alternative to the SGML-structured queries supported by Pat. For example, the APIB gateway allows users to search for EXPRESS entities (or other schema objects) by specifying a substring of the object's name. It would be desirable if the APIB gateway also allowed users to search for EXPRESS entities by specifying a natural language description of the entity. PRISE could match this query against each entity's ENTITY.DESCRIPTION element (Figure 10 shows the ENTITY.DESCRIPTION element for the entity `dimension_curve_directed_callout`).

## 5.2 Gateway for Accessing Humanities Texts

A gateway for accessing humanities texts has been implemented that, like the APIB gateway, provides a World Wide Web interface for accessing information in SGML documents using Pat as a retrieval engine. The documents include novels, poetry, dictionaries, and other literary and reference works tagged in SGML according to the Text Encoding Initiative's (TEI) *Guidelines for the Encoding and Interchange of Machine-Readable Texts*. This gateway is currently being used by the University of Virginia library and the University of Michigan's Humanities Text Initiative for dissemination of texts to their respective university communities and to the general public. The gateway's CGI scripts are written in perl.

This CGI application is similar in many ways to the APIB gateway. Indeed, its development predates that of the APIB and helped influence the APIB's implementation. The differences between the gateway for accessing humanities texts and the APIB gateway stem mainly from differences in their respective DTDs. Because the DTDs used for representing the humanities texts are not as rich in SGML elements as the STEP DTDs, the possibilities for structured queries available to users are not as extensive. On the other hand, the TEI Guidelines specify sophisticated linking mechanisms that support hyper-linking to and retrieving a chunk of text even if the text is untagged in the document. One of the ways this mechanism is being employed in the gateway is in a CGI program that allows users to specify a range of line numbers in a poem and generates a web page containing the text from the lines specified[AMV]. This application of TEI linking provides a convenient way for writers to cite passages in the gateway's texts.

## 5.3 The Digital Library Initiative

The US Digital Library Initiative (DLI) is a federal program funding research at six universities to further the state of the art in searching for and displaying selections from large, heterogeneous collections of on-line reference materials. The DLI project at the University of Illinois is building a testbed for querying multiple SGML repositories of scientific publications over the Internet as if they were organized into a single collection. This process of merging the diverse text collections into a single virtual repository is called "federation". Since the document collections use different DTDs, the project uses a canonical DTD based on the ISO 12083 Article Document Type Definition. Each collection is translated from its original DTD into the canonical DTD for indexing and retrieval. Pat is used as the search engine for the federated repository.

The DLI testbed's federation approach could be useful for merging the APIB with other repositories of international standards. ISO has developed its own DTD for representing international standards documents. This DTD could serve as the canonical DTD for a federation of international standards collections. However, because the ISO DTD is not specific to any of the collections, an ISO-federated repository would not support searches based on specific structures of any particular collection.

SGML-tagged query results are displayed using Panorama[SQ], a commercial SGML viewer, rather than being translated to HTML for display with a web browser. The DLI testbed's implementation includes a mapping between elements in the canonical DTD and display styles used by Panorama. This mapping is analogous to the APIB gateway's SGML-to-HTML translator. Because query results are displayed in their native SGML rather than in HTML, they can include items HTML is incapable of representing such as complex mathematical equations. Also, because Panorama supports some of the HyTime[HYTIME] international standard (ISO 10744), an application of SGML describing the structure of documents for use in hypertext and multimedia applications, the DLI testbed can support a richer variety of hyper-links between documents than an HTML browser alone can support. However, users need to have HyTime-aware SGML viewing software installed locally in order to use the DLI testbed.

In addition to the testbed, the University of Illinois DLI project is conducting research in several other areas. These include integration of multiple query engines and information retrieval from digital libraries based on deep semantics[8]. Another area of research is in techniques for keeping track of state information for federated CGI applications. The

[8]This involves using natural language understanding techniques from artificial intelligence to semantically analyze the texts in the digital library and to analyze queries.

University of Illinois DLI project (along with the other DLI projects) has the potential to make a major contribution to the global infrastructure for on-line information retrieval over the next several years.

## 6 The Future

The APIB gateway currently supports browsing and querying of IRs and partial browsing of APs. It contains only a subset of the initial release of STEP. There is a need both to expand the collection of documents accessible through the APIB gateway and to provide a wider variety of services to users. The IRs and APs currently available were converted into SGML manually from documents originally written using commercial word processing software. Although conversion of legacy data into SGML is a long and tedious process, efforts are underway to tag the remaining existing STEP IRs using the NIST-developed DTD[9]. Tagging the rest of the IRs in the initial release of ISO 10303 is the highest priority. The next highest priorities are tagging IRs that have attained Draft International Standard (DIS) status and tagging the remaining APs in the initial release. In the mean time, current AP developers are being encouraged to use a NIST-supplied environment[PHIL] for authoring new STEP AP and IR documents in SGML. In the near future, the APIB gateway's functionality will be extended to allow users to access all components of SGML-tagged STEP IRs and APs. This will require extending the APIB's data access services to support information retrieval from AP documents using the full set of structures defined in the AP DTD.

Plans also call for expanding the scope of the APIB and the APIB gateway to include documents created during AP development that are separate from the STEP AP document. Examples of these documents are issue logs, status summaries, and validation reports. Error reports and enhancement requests resulting from implementation experience may be included as well. These documents may not be represented in SGML but nevertheless will need to be associated with elements in the SGML-tagged APs and IRs in the APIB. Thus the APIB gateway will have to support hypertext browsing between SGML and non-SGML documents. More sophisticated linking mechanisms such as those specified in the TEI Guidelines (Section 5.2) or in HyTime (Section 5.3) may be desirable for this purpose.

In addition, efforts are underway to provide users with access through the APIB gateway to tools for creating EXPRESS models. This will be done by providing a World Wide

[9]The legacy IRs and APs were written using either commercial word processing tools or LaTeX[LAMP]. An auto-tagging software tool has been implemented to help convert the LaTeX IRs into SGML, but this tool assumes that the IR document uses a particular set of LaTeX style files developed for STEP[WILSON]. Unfortunately, some of the legacy LaTeX IRs use LaTeX customizations that are incompatible with the auto-tagger.

Web interface to the NIST EXPRESS Server[LIBES94], a service that allows users to run EXPRESS software tools remotely without installing them locally. The EXPRESS Server currently uses electronic mail for its user interface. Users send requests and EXPRESS code to the Server as e-mail. The output from running an EXPRESS tool can then be e-mailed back to the user, or it can be stored in the user's private directory on the Server. Once the EXPRESS Server is made accessible through the APIB gateway, STEP AP developers will be able to build EXPRESS models re-using IR constructs by querying the APIB for the desired constructs and feeding the results directly into an EXPRESS tool. Among the tools available through the EXPRESS Server are an EXPRESS syntax checker as well as a tool for converting an EXPRESS schema with external references (see Figure 4) into a self-contained EXPRESS model[LIBES93]. This latter tool is very useful for building the "system integrator's view information model" discussed in Section 3.2 because this EXPRESS model needs to be self-contained so that an AP implementor can use it as a specification without having to refer to additional documents in ISO 10303.

## 7 Concluding Remarks

Section 1 mentioned that the APIB serves as the central information registry for an integrated software tool suite for AP developers. This tool suite, the Application Protocol Development Environment (APDE), is already having an impact on the development and implementation of STEP APs. Portions of the APDE that are presently operational are being widely used by AP developers and implementors. For example, the EXPRESS Server receives hundreds of requests per month for remote execution of NIST-developed tools, and use of the tools is resulting in shorter AP development times[SAUDER]. Also, ambiguities and inconsistencies in the STEP and ISO documentation guidelines have been exposed through the effort of developing the STEP DTDs[PHIL]. Since the STEP DTDs provide a complete and unambiguous specification of AP and IR document structure, AP and IR documents authored in SGML will be free of the inconsistencies and documentation guidelines violations that plague the APs and IRs in the STEP initial release.

The APIB gateway provides a convenient way for AP developers to access the existing body of information in STEP. The APIB gateway combines the representational power of SGML with the ease of electronic delivery of HTML, resulting in an easy-to-use system that is robust and extendible. It appears to users as if they are "connected" to the APIB at all times. In reality, they are only using APIB services during the brief intervals between when they issue CGI requests through HTML forms and when the APIB's retrieval engine returns the data necessary to generate a web page. Therefore, the APIB gateway can easily accommodate many users at a time. As the variety and number of items in the APIB continues to grow, the APIB gateway will become an increasingly valuable resource to the STEP community.

The STEP documents in the APIB are copyrighted by the International Organization for Standardization (ISO). Therefore, access to the APIB Gateway is limited to developers of ISO standards. Requests for further information should be directed to the author.

This document was written in SGML using the DocBook DTD[DOCB]. It was translated to LaTeX[LAMP] using the NSGMLS and SGMLSpm software tools mentioned in Section 4. The author thanks this document's NIST reviewers — Mary Mitchell, Don Libes, and Paul Over — for their comments and corrections.

## References

[AMV] University of Michigan Humanities Text Initiative and University of Michigan Press, *American Verse Project*, 1996.
URL: http://www.hti.umich.edu/english/amverse/

[BARN] Allison Barnard Feeney, Mitchell Gilbert and Yuhwei Yang, *Guidelines for AIM Development*, ISO TC184/SC4 Technical Report, N435, April 5, 1996.
URL:
http://elib.cme.nist.gov/msidsubject/sc4/howto/methods/ap/ballot1/sc4n435.ps

[BLEE95] Tim Berners-Lee and Daniel Connolly, *Hypertext Markup Language - 2.0*, Internet Engineering Task Force, HTML Working Group, RFC 1866, November 1995.
URL: ftp://ds.internic.net/rfc/rfc1866.txt

[BLEE96] Tim Berners-Lee, Roy Fielding and Henrik Frystk Neilsen, *Hypertext Transfer Protocol – HTTP/1.0*, Internet Engineering Task Force, HTTP Working Group, Internet Draft, February 19, 1996 (expires August 19, 1996).
URL: ftp://ds.internic.net/internet-drafts/

[CLARK] James Clark, *SP*, version 1.0.1.
URL: http://www.jclark.com

[DOCB] HaL Computer Systems, Inc. and O'Reilly & Associates, Inc., *Guide to the DocBook DTD*, version 1.0 for release 2.2.1, February 24, 1995.
URL: http://www.ora.com/davenport/README.html

[GOLD] Charles Goldfarb, *The SGML Handbook*, Oxford University Press, 1992.

[HAR] Donna Harman and Gerald Candela, *Bringing Natural Language Information Retrieval Out of the Closet*, SIGCHI Bulletin, Volume 22, Number 1, July 1990.

[HYTIME] Fujitsu Open Systems Solutions and TechnoTeacher, Inc., *HyTime Application Development Guide*, Version 1.2.4, February 1996.
URL: ftp://ftp.techno.com/pub/HyTime/

[ISO1] International Organization for Standardization, *Industrial automation systems and integration—Product data representation and exchange—Part 1: Overview and fundamental principles*, ISO 10303-1, 1994.

[ISO11] International Organization for Standardization, *Industrial automation systems and integration—Product data representation and exchange—Part 11: EXPRESS language reference manual*, ISO 10303-11, 1994.

[ISO101] International Organization for Standardization, *Industrial automation systems and integration—Product data representation and exchange—Part 101: Integrated application resources: Draughting*, ISO 10303-101, 1994.

[LAMP] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, second edition, 1994.

[LASK] Sharon Laskowski and Venkata Ramayya, *Electronic Access To Standards On The Information Highway*, National Institute of Standards and Technology, NISTIR 5708, August 1995.
URL: http://dsys.ncsl.nist.gov/nssn/index.html

[LIBES93] Don Libes, *Shtolo - Converting STEP Short Listings to Annotated Listings*, National Institute of Standards and Technology, NISTIR 5291, November 1993.
URL: http://elib.cme.nist.gov

[LIBES94] Don Libes, *Concepts of the NIST EXPRESS Server*, Proceedings of *First International Workshop on Services in Distributed and Networked Environments (SDNE)*, Prague, Czech Republic, June 27-28, 1994.
URL: http://elib.cme.nist.gov

[LIBES96] Don Libes, *Writing CGI Scripts in Tcl*, Proceedings of *Tcl/Tk Workshop 96*, Monterey, CA, July 10-13, 1996.

[LUB95] Joshua Lubell, *Patparse*, National Institute of Standards and Technology, initial release.
URL: http://elib.cme.nist.gov/msidsubject/sc4/tools/nist/patparse.tar

[LUB96] Joshua Lubell and Lisa Phillips, *Application Protocol Information Base Architecture*, National Institute of Standards and Technology, to appear, 1996.

[MEGG] David Megginson, *SGMLS.pm: A perl5 class library for use with the SGMLS and NSGMLS parsers*, University of Ottawa, version 1.03.
URL: ftp://aix1.uottawa.ca/pub/dmeggins/

[OTC] Open Text Corporation, *System Integration Guide*, Release 5.0, 1994.

[OUST] John Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

[PALMER] Mark Palmer and Mitchell Gilbert, *Guidelines for the Development and Approval of STEP Application Protocols, Version 1.2*, ISO TC184/SC4 Technical Report, N433, April 5, 1996.

[PHIL] Lisa Phillips and Joshua Lubell, *An SGML Environment for STEP*[10], National Institute of Standards and Technology, NISTIR 5515, November 1994.
URL: http://elib.cme.nist.gov

[PWIL] John Price-Wilken, *Using the World-Wide Web to Deliver Complex Electronic Documents: Implications for Libraries*, The Public-Access Computer Systems Review, Volume 5 no. 3, 1994.
URL: http://sansfoy.hti.umich.edu/www-to-pat/

[ROGERS] Willie Rogers, Gerald Candela and Donna Harman, *Space and Time Improvements for Indexing in Information Retrieval*, Proceedings of *4th Annual Symposium on Document Analysis and Information Retrieval*, University of Nevada at Las Vegas, USA, April 1995.
URL: http://potomac.ncsl.nist.gov/~rogers/papers.html

[SAUDER] David A. Sauder, Mary J. Mitchell and Allison Barnard Feeney, *Challenges to the National Information Infrastructure: The Barriers to Product Data Sharing*, National Institute of Standards and Technology, NISTIR 5498, September 1994.

[SCHATZ] Bruce Schatz, William H. Mischo, Timothy W. Cole, Joseph B. Hardin and Ann P. Bishop, *Federating Diverse Collections of Scientific Literature*, Computer, Vol. 25, No. 5, May 1996.
URL: http://www.grainger.uiuc.edu/dli

[SQ] SoftQuad Inc., *Panorama*, version 1.5.
URL: http://www.sq.com

[VROM] Johan Vromans, *Perl 5 Desktop Reference*, O'Reilly and Associates, February 1996.

[WILSON] Peter Wilson, *The LaTeX Package Files User Manual*, ISO TC184/SC4 Technical Report, Draft, July 1995.
URL:
http://elib.cme.nist.gov/msidsubject/sc4/editing/latex/current/isosty.ps

[10]Also appeared in *SGML '94 Proceedings*, Graphic Communications Association, Vienna VA, November 1994.